

Set by Seyb
PMK 11 Jan 69

D A T A T E X T I N P U T

i n t o

T H E R O C A P P I S Y S T E M

Rocappi, Inc.

Second Edition: June, 1969

*A Division of The Lehigh Press, Inc.

7000 North Park Drive, Pennsauken, N.J., 08109

I N T R O D U C T I O N

The Datatext system, made available by the IBM Service Bureau Corporation, offers to the user an on-line text-editing capability, providing ready access to his manuscript or "file," ease in making corrections, and producing an upper and lower case printout, either through the terminal itself, or, overnight, by on-line printer.

The final draft of the manuscript can be formatted to a desired depth and measure (line length) and aligned right-hand margins can be provided, if specified, since the Datatext program has the capability of "justifying" each line through the addition of extra interword spaces. (The program does not have the capability of "hyphenating"--that is, making word divisions at the end of lines; nor are the type characters proportional). It is in part

for this reason that the additional step--photocomposition-- is a desirable objective and an interface between Datatext and computerized photocomposition is therefore sought.

When the manuscript has been reviewed and is found to be error free, it is then possible to make available to Rocappi, Inc., a magnetic tape which will serve as input to the Rocappi System of Book Composition and which can be processed, as well, through many other Rocappi programs, such as for sequencing, abridging, "exploding entries," and the like.

The Insertion of Codes or Symbols

Any system of computer composition, such as Rocappi's, requires the addition of certain codes or symbols which are extraneous to the "text stream" of the manuscript itself, and which are, to some degree at least, distracting to the reader. The number, variety and frequency of such symbols depends entirely upon the demands which will be placed upon the "data file" or manuscript. Few such symbols will be required if the text is straightforward and makes little use of the typographic capabilities of the Rocappi System. If extensive changes in type face or format are

desired, if an extended character set is required, or if the manuscript is to serve as a file or "data bank" for information retrieval and manipulation, the inclusion of such "flags" or symbols must of necessity be more extensive.

Even for the simplest kind of composition--the novel, for example--some typographic conventions are ultimately required, although they need not be inserted in the initial Datatext keyboarding of the manuscript. Several different techniques are possible, depending upon ease of access to Datatext keyboards, degree of familiarity with the Rocappi System, and alternative uses for which the file may be intended.

Here are some of the alternatives which suggest themselves:

1. Initial input as an editorial function, and updating for style consistency and copy-fitting, then turnover of file to Rocappi for insertion of all typographic codes and symbols.
2. Initial input as an editorial function, review and updating for style consistency and copy fitting, and insertion of some of the typographic codes at that time. Turnover to Rocappi for insertion of more complex

codes and symbols, either through Datatext or through conventional Rocappi correction procedures.

3. Insertion of many of the required typographic codes and symbols at the time of initial input. After updating and review, final insertion of remaining codes or symbols prior to the turnover of the file to Rocappi for entry into the Rocappi System.
4. Initial input by the author, using the Datatext system as a convenience in the preparation of the original manuscript itself; further processing of the same file by the editorial department, including review for style consistency, copy-fitting, and perhaps insertion of typographic codes. Simultaneous review may take place by the indexer or for purposes of preparing an abstract, or in order to provide key words for information classification and retrieval.

It will be seen that there are many variations possible, and that an approach which is appropriate to

one situation may be less so in another. Central to our concept, however, is that, if you so desire, several people can have access to the same file, including Rocappi's own mark-up and planning staff. Rocappi has installed its own Datatext terminal expressly for this purpose. All that is necessary is to safeguard the initial file, by assuring that others who work on it will process it for separate storage, without destruction of the master file.

Access to the file is facilitated by the fact that the central processor serving the eastern seaboard is located in Philadelphia. When more such processors are required, communication between these different locations may possibly be arranged.

The first section of this manual is devoted to an explanation of the typographic codes or symbols referred to above. When they are well understood by the reader he will be able to determine for himself the extent to which he wishes to assume responsibility for inserting some, or all of them, and the stage in the creation and development of the file when it seems most convenient to add them.

SECTION 1: SYMBOLS OR CODES REQUIRED FOR TYPESETTING

Typesetting requires the use of two categories of spaces: variable and fixed spaces.

The Justifying Space. The normal interword space is produced by the ordinary spacebar at the bottom of the keyboard, or by the use of the carriage return key. The use of the carriage return key does not in fact insert a space, but this is accomplished by program because in another version of that same text the lines may not--probably will not--break in the same fashion. Hence the operator may assume that, where required, an ordinary space will be added at the end of such lines.

Interword space of the type we are describing is called "justifying space," or spaceband space. This is because, for typesetting, this space is variable in width. (Linotype machines use spacebands to achieve this variability in space width.) In computer composition the width of these variable spaces is calculated by computer program, in order to justify the line to the desired measure. Hence when a variable or justifying space is not intended the operator must so specify.

Fixed Spaces. Normally, at least three types of fixed spaces are necessary for typesetting. They are:

- a.) the em space
- b.) the en or "nut" space
- c.) the "thin" space

Each of these spaces is relative to the size of type being composed. But since they are fixed spaces their widths will not be varied in order to help make the lines justify.

Normally, one or more em spaces is used at the beginning of paragraphs, because the amount of indention to indicate paragraph openings should be constant, at least in relation to the point size of the type, and should not vary according to the tightness or looseness of the line.

There are times when an en space or a thin space will be desired, as for example, when lists are being prepared, and a fixed unit of space will be specified in the manner shown below:

±01.)#Here.

±02.)#And here.

±10.)#And here.

The space after the parenthesis is a fixed space, usually an en or a thin. The space before the numbers is also a fixed space, usually one or more ems. In this case an additional en space is used to align the figures of one digit with those of two digits.

Rocappi sometimes uses a fourth kind of fixed space--a variable fixed space. This is most often used for letterspacing, when a caption is to be set in all caps, and the customer desires the individual characters within the caption to be slightly separated from each other.

NOT LETTERSPACED

L E T T E R S P A C E D

An em, en or thin space between such letters may appear to be too much. If the variable fixed space code is used a smaller increment of fixed space may be inserted in typesetting, and the amount of such fixed space may be varied upon demand, in order to achieve the best effect. Because of the limitations of the Datatext keyboard it is sometimes necessary to improvise solutions, and in this instance an overprinted character generates a new code. Use the letter h, followed by a backspace, and an overprint of the number 1, as follows:

~~L E T T E R S P A C E D~~

Since the Datatext keyboard is limited, it has been necessary to sacrifice certain keyboard positions in order to obtain the typographic codes required to indicate fixed spaces. As shown above, the "upshift" 1, which prints out as ±, is now pre-empted for the em space. The upshift 2 (a) is pre-empted for the en space, and the upshift 3 (#) is used for the thin space. The variable fixed space, since it is used less frequently, is achieved by striking a combination of three keystrokes and therefore does not require sacrificing any keyboard position. As shown this is done by striking the letter b, followed by a backspace, and then overstriking with the 1. It is obvious that the attention key is not struck prior to the backspace.

Summary of space codes

Justifying space: the space bar or the single carriage return.

Em space: upshift 1, which is ±

En space: upshift 2 or a

Thin space: upshift 3 or #

Variable fixed space: b backspace 1, prints out as ½

If the customer does not feel that these space codes are too distracting, it is recommended that they be inserted in the initial input. Hence, if a two-em paragraph indentation were desired, for example, the hard copy or the printout of the on-line printer would look like this:

++This is the beginning of a paragraph with a two-em indent.

The Problem of Quotation Marks

In type, there is a difference between a left and a right quotation mark, whether single or double. The left mark has a "6" look about it, and the right mark might be called a "9." By convention, then, the opening double quote can only be generated by two keystrokes in the unshift position, and the closing double quote by two keystrokes in the downshift position.

""It is time for you to come home now,"" she said.

There is no way around this inconvenience unless the customer wishes to order a separate golf ball which is re-designed with single opening and closing quotes.

If single quotes are followed, the same concept applies:

"'When I say 'come' I mean it.'"

Not All Key Positions Are Available

Certain of the "upshift" number positions cannot be accessed in the keyboard family layout which we call "D-1." Please consult the section of the manual dealing with keyboard layout families. For example, on our Datatext keyboard the upshift 6 shows a square bracket. It is only a left bracket and no right bracket obtains. Reference to the keyboard layout will show that there are other uses for this key in certain cases, and that brackets are created in other ways.

Line Ending Codes

For typographic reasons, it is necessary to identify the following situations:

1. An end of paragraph, or a single line, which is to be flushed left. This means that the next sentence or entry is never run-on. It is essential to signify this

condition, illustrated in this very paragraph, by the use of the "+". The "upshift" equal sign has thus been pre-empted to mean a "quad left" code. It must be used at the end of every paragraph.+

2. An instruction to center a single word or group of words. Use the same code twice.

Illustration:

CENTERED++

THREE WORDS CENTERED++

3. An instruction to flush right a single word or group of words. Use the same code three times.

Illustration:

FLUSH RIGHT+++

4. An instruction to space out between one word or word group and another, so that the first word or first group is flushed left, and the second word or second word group is flushed right in the same line. Use the equal sign followed by a period and a space. Since the space (a "justifying space") does not print, we have represented it here by a bracket. Please remember that it is a regular space and not a bracket.

ILLUSTRATION:

John Smith+. [

KI 4-1155+

Note that a quad left code is used at the end of the line.

Leadering.

If, instead of blank space between the left portion and the right portion, a row of periods (leaders) is desired, this may be accomplished by the use of =... (equal period period period.)

If you want to alternate the period with a space to provide a less solid leader, alternate the period and some kind of fixed space. For example, =.#. would alternate periods and thin spaces. If you called for =.-. you would alternate periods and hyphens.

A solid horizontal rule can be created by using this same code, in the proper type family (where a solid full-length rule of the proper weight will be found) and then by perforating "=. rule rule" (we use the word "rule" in this explanation merely to indicate the appropriate key for that particular rule).

The most common leader will consist of periods and thin spaces, and it is usually better to put the thin space first, although this is a matter of preference.

In other words, the =. code says, "fill up with something." If the third character is a justifying space, then the line is to be filled with blank space. If the third character is not a justifying space, then there must also be a fourth character, and the area will be filled with an alternation of those two characters, both of which could be periods.

Type Face Identification

You will have noticed that we have pre-empted the "=" sign key, using the upper position for a quadding code, and the lower position for a different kind of precedence or function code, which we call a "Beta." Hence the "=" sign may not be used to mean equals. If you wish to use the true character consult the discussion of type families and extended character sets.

The "=" sign is a flag or prefix which alerts the computer program to the fact that an instruction will follow. The next character identifies the instruction. If the instruction consists of any of 25 lower-case characters, a- through z, except for x, the two characters in conjunction ("=a" for example) make up a tag or label

to identify the text which follows as to type face, font or family.

Example: Let t stand for the text type face, and i stand for the italic face associated with the text type.

=t "Please come here =iat once!" =the said.

=t "Please come here =i at once!" =t he said.

You will notice there are two options. Either you can affix the two-letter code to the next word, or you can use a space following the two-letter code. The Rocappi program will eliminate that redundant space.

If we take into account the fact that there is an em indent at the beginning of the line, the actual representation would be:

=t± "Please or

=t ± "Please

The Datatext user will have to decide whether these codes will be confusing to the reader, if the manuscript is to be seen as Datatext output by persons not familiar with the requirements of typesetting. If he feels they would be distracting he may choose to add them later. If they are not too frequent, they may not intrude too much.

A signal or flag to designate a type face governs until it is over-ridden. Hence if you are in the text type face and wish to remain in that face for some considerable period of time it is not necessary to repeat the instruction. It is only when you wish to go in and out of a condition, such as italic, that you must so indicate.

However, if all changes in type face are invariably associated with changes in format you can avoid the use of such coding, and incorporate the text and other type face instructions as part of generalized "format" or "macro" codes.

But there is no way to avoid the use of such codes when type face or family changes occur in the middle of a text stream, all of which is in a given format.

How to Define Type Faces

You do not need to decide in advance what type faces or sizes you will require. All you need to know is that you will (or may) want different faces or sizes to serve different purposes and should "tag" them accordingly. For example:

My text, which I will later choose to be either Times Roman or Baskerville, perhaps, may be 10/12 or 8/9, or sometimes one and sometimes the other. However I later define it I will call it "=t" because it is the basic text face.

The italic which will go with it I will call "=i."

The extract, which will be roman, and perhaps one size smaller than the text, I will call "=e" and if there is an occasional italic in the extract, since I have already used "=i" I will call it "=j."

The bold which will go with the text I may choose to call "=n."

The small point size for footnotes I will call "=f."

Later, I may decide that I do not want extract, so I can define "=t" and "=e" to be the same face and size.

The Rule in Designating Type Faces. When you define a type face it continues to be associated with the text until you specify otherwise. If you go into italic and forget to come out of it the entire manuscript from that point on would remain in italic. You come out of it by:

specifying another type face, or

entering into a new general format statement

which also implies an associated type face.

Use of Format Statement

If you wish to define elements of composition you may simplify the keyboarding operation and eliminate a number of codes which might otherwise need to be contained in the text. What are these elements?

They might be:

1. Chapter Opening, including instruction for

placement of page on which chapter starts,
amount of sinkage, position of chapter heading,
type face and style in which it appears. For
convenience, let us designate a chapter
opening element as co and flag it as =\$co.

2. Begin chapter text, following chapter head.
There will be some "leading" after the head,
and then the text will begin. Call this ct
and flag it as =\$ct.
3. Number 1 head in text. You will need to
identify the beginning and ending of this
head since leading will be associated with
each. Call this ha and hb for head one, hc and
hd for head two, and he and hf for head type #3.

Example:

```
=$ha CAPTION TYPE ONE=$hb  
=$hc Caption Type Two=$hd  
=$he caption type three=$hf
```

4. Extract may require space before it, an indent
condition, such as a center indent or a left,

and a smaller type face. You can call this be for begin extract and perhaps use bf to signal the end of the extract.

The flags would thus be =\$be and =\$bf.

The Use of "Beta Codes"

The = sign is used as a function code or flag in the Rocappi System. It has been given the name of "Beta" and is normally a special key on the keyboard, with its own character or hard copy image. However, on the Datatext keyboard this key does not exist, and hence the "=" sign has been pre-empted to serve the same purpose and it may be called "Beta."

When the program encounters the Beta code it examines the next character:

if the next character is an x, three digits will follow which define the line length or measure.

=x240 means a measure of 24 picas and no points.

=x246 means 24 picas and six points.

The Beta code is used for other purposes. You have already encountered its use to flag type faces and to set forth format statements. (=\$aa for example). Other functions of the Beta code will now be discussed.

Setting and Changing the Measure

The first information to be set forth in the input stream should be the initial measure of setting, unless this information

is contained in a format statement.

The measure message can be:

attached to the first word of the input, or
can precede the first word, being separated
by an ordinary (justifying space) or a single
carriage return.

Examples are shown on the following page.

Examples:

=x240=j+This is the first paragraph.

or

=x240=j +This is the first paragraph.

or

=x240

=j

+This is the first paragraph.

(Note that a type face also must be defined before the first word of input, unless it is included in a format statement.)

The length of the measure can be changed at any time. It is usually changed at the beginning of a paragraph. If it is changed in the middle of a line it will apply at the beginning of the next line.

Adding "One-Time" Leading

The Beta Code followed by an apostrophe and three digits will call for film advance in points, to be effective at the beginning of the next line. It is called "One-Time Leading" because this additional amount of leading will only apply for that particular line. This code is usually used at the beginning of a paragraph or before a heading or sub-head. The leading instruction may of course be included in a format statement. If it is directly coded it might appear as follows:

end of paragraph of text.

= '030CAPTION

= '015New Paragraph

Up to 999 points of leading can be added in this way.

= '001 is add one point of leading.

= '010 is add ten points of leading.

= '100 is add 100 points of leading.

In the application of pagination, leading added in this fashion can be converted into variable leading--that is, the amount of leading called for may be increased or diminished somewhat to effectuate satisfactory pagination.

The Concept of Justification

Normally running text will be justified. Justification will take place for all lines other than those for which a "quad code" (left, right or center) governs. When the quad condition governs, the lines will not be spaced out. Otherwise, unless justification is cancelled, space bands (interword spaces) will expand to fill up the line and give it even margins.

Sometimes it is desired not to permit the interword spaces to expand. This will result in ragged setting, and it is necessary to determine whether the raggedness should appear on the right, on the left, or equally on each side of the measure.

=;l(ell) calls for an even left margin and a
ragged right.
=;r calls for an even right margin and a ragged left.
=;c calls for a centering of each line.
=;j returns to the justified condition.

These codes may be included in format statements or used whenever desired. They are usually found at the beginning of paragraphs or blocks of texts. Where heads are to be set it is always desirable to go into ragged setting.

Illustration

=;l=m THIS IS A HEADING AND IT

MAY TAKE UP TWO LINES

=;j=r+Now we have returned to justified setting.

Relation of Hyphenation to Justification

Often it is desirable to kill the hyphenation assumption when setting ragged, particularly for heads (although running text is frequently set ragged with hyphenation).

- =-0 kills hyphenation, and should be used to precede ragged heads.
- =-1 also kills hyphenation, but will permit text to break on hyphens or em dashes (as well as em and en spaces).
- =-2 returns to hyphenation, except that the last word of a paragraph will not be broken.

This kind of hyphenation will not happen.

- =-3 returns to hyphenation, and will permit hyphenation of the last word of a paragraph.

Illustration

=;l=-0=m THIS IS A HEADING AND IT
MAY TAKE UP TWO LINES OR MORE,
BUT WILL NOT PERMIT HYPHENATION

=;j=-3=r⁺Now we have returned to justified and hyphenated setting.

The quality of justification will depend upon frequency of hyphenation. Frequency can be strongly influenced by the amount of interword space expansion allowed by parameters to the Phase II program. Ask for further information on this point.

Handling Indentions

Perhaps the most complicated input problems are associated with indents and tabular material. This section will discuss indents.

General Format of Indent Statement

When the computer encounters the first Beta flag it looks at the next character, discovers it is also a Beta and knows that the instruction pertains to indents. The third character defines the kind of indent (hanging, left, right, center or variable).

==l (ell) is a left indent

==r is a right indent

==c is a center indent

==h is a hanging indent

==v is a variable indent

The next three digits define the amount of indent, in ems and ens of the point size of the defined type face.

If I am in =r, and the type face is to be defined as ten point, then

==r010= calls for a right indent of one 10 pt. em
==r110= calls for a right indent of eleven ten pt. ems
==r001= calls for a right indent of one 10 pt. en
(In other words, the third digit is ens)

Similarly,

==c020= calls for a center indent of two ems.
(In the case of a center indent, the amount of space called for will appear on both sides of the indent.)

It will be noted that a Beta code ends the indent instruction. This is true if there is only one indent instruction. In case there is more than one (see "nested indents") the manner in which such indents is terminated will be described below.

Format of Indent Message (Indefinite Duration)

==cthree digits= if the indent is to continue until
cancelled. (c or l, h, or r)

Format of Indent Message (Definite Duration--Self Cancelling)

==cthree digits plus two digits=

==c03010= where first three digits state amount of indent
(three ems) and last two state number of lines
for which indent is to continue.

Cancelling Indent

No need to cancel an indent which obtains only for a stated
number of lines (indent of definite duration).

Other indents must be cancelled.

Cancelling is best achieved by using +=,
but other levels of indent are cancelled by +=a, +=b,
+=c and +=d.

Nesting Indents

Five levels of indents can be handled in nested form. They are
identified as follows:

First indent is called the Beta indent

Code == { l
 r
 c
 v
 h } 030= , cancelled by +=

or == { l
 r
 c
 v
 h } 03010= , self cancelling

Second indent is the "a" indent

Code
== (l, r, c, v or h) 030a , cancelled by +=a
or self-cancelling.

Third indent is the "b" indent

==(l,r,c,v or h)030b , cancelled by =+b
or self-cancelling.

Fourth and Fifth Indents are c and d indents, and are
coded and terminated as above.

However, =+= terminates not by the first or Beta indent,
but all other indents in effect at that point.

When to use each type of indent.

Hanging indent is a left indent which does not apply to first
line of a paragraph.

```
-----  
-----  
-----  
-----  
  
-----  
-----
```

Input ==h030= (followed by space, one carriage return, or
first word of text).

==h030=

This is the beginning of the hanging
indent, which is of three ems in depth,
and of indefinite duration.

Each paragraph starts at the left margin
again.

++Of course, the first characters can be
spaces.

+l Or spaces and numbers. =+=

Variable indent is also a hanging indent, but the amount of
indent must be calculated by the program, and hence three
zeros are put into the statistics. The program replaces
these zeros with the appropriate values. The indent code
must be inserted at the precise place where the calculation
is to be made. Do not use spacebar code to left of indent
code.

++l.)#==v000=The amount of indent is equal
to the width of two em spaces, the
number, the period, the paren, and
the thin space (upshift 3) which
follows.

The left, center and right indents require no further explan-
ation.

Illustration of Nesting

```

      .....measure.....
      'xxxxxxxx text xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
==c030=      'xxxxxxx center indent xxxx'
              'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
              'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
              'xxxxxxxx.'
==l020a      'xxxxx left indent xxxxx'
              'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
              'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
==r010b      'xxxx right indent x'
              'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
              'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
              'xxxxxxxxxxxxx'
              'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
              'xxxxxxxxxxxxxxxxxxxxxxxx'
==+a
              'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
              'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
==+=
      'xxxx back to full measure again xxxxxxxx'

```

"Freezing" an indent

If the amount of indent calculated as a variable needs to be converted into a left indent, it can be "frozen" by calling for a "dummy" indent of zero width value. This is because each indent picks up the value of the preceding depths of indent and adds its own value to it.

```

xxxxxxx measure xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
t++1.)#==v000=Hanging indent requires
              that first line in paragraph
              revert to margin.
==1000a t++But here we wish to indent
              a new paragraph, and so we
              must convert the hanging in-
              dent into a left indent.

```

Use of Macro Codes for Indents

Often it is desirable to simplify formats by the use of the =\$ format codes or "macros" (since it is possible to substitute for these "=\$" codes any kind of information, whether of a text character or coding).

Hence =\$ia
=\$ib
=\$ic could describe three standard indent conditions for a job.

Use of Shorthand Input Techniques

The "=\$" codes are shorthand codes, and it is possible to substitute text material or coding instructions for these codes. Often they may be used for format information, especially for tabular material.

The Rocappi System is capable of setting highly complex tabular material, but it is most useful to describe this capability in a separate manual. Ask for it if you need it. You will find the "=\$" codes especially useful in the input of tabular material.

But you may also find them useful for handling complicated re-iterative material of a text nature. For example, if you find frequent repetition of a term such as Na_2Cl_3 you may find it convenient to assign a "=\$" code. This can be done in two ways:

The first time you encounter this term and input it you can put ==\$a on either side of it =\$a as shown here. Then the times after that when you encounter it you can input =\$;a and wherever this coding occurs the material between the ==\$a codes will be regenerated.

Or, if there is more than one such shorthand text condition that you want to use, then assign =\$ codes such as =\$aa or =\$ab and be sure to tell us what they are supposed to mean. (A transmittal form can be devised.)

Use of Beta functions ("=") for Keyboard Layouts.

We have already stated that a different Beta flag should be used for each variation in type face, size or family.

Variation in face

a change from =t to =i for going from roman to italic in the text face (such as Times Roman 10/12)

Variation in face and size

a change from =t to =d, from Times Roman roman 10/12 to Helvetica, light, display, 18/18, for example.

Variation in face but not size

a change from =t to =b, where bold is to be in same size as text face, but sans serif. A change from Times Roman 10/12 to Helvetica bold 10/12, where Helvetica is to be used as a run-in head.

Variation in family

a change to pick up a "peculiar"--a pi character, or a Greek character, or a small cap, or a diphthong. (see keyboard layout sheets). These require a new Beta function for each point size.

Use of superiors and inferiors

While the same keyboard layouts can be used, a different Beta flag is required to indicate that they are to be positioned as superiors or inferiors.

A superior which is related to one-size type must be distinguished from a superior which is related to another size type.

Keyboard Family Layouts

For Fototronic Output via Datatext

D1 is standard layout

D2 makes available Old Style numbers, cap and small cap in roman, diphthongs in roman, cap and small cap, and one or two additional characters.

D3 provides floating accents for use over any characters in the D1 or D2 layouts.

D4 is the Greek family layout.

D6 provides "peculiar" or "pi" characters, as well as math symbols, and various weights of horizontal or vertical rules.

Can be used in Superior & Inferior Positions

Rho = 8 B
Delta = 8 C
Gamma = 8 F

IBM DATATEXT

FAMILY

D 1 (for Fototronic CRT Output)

B1	B2	B3	B4	B5	B6
Var. W/S	EN Dash		[]	[]	

ONE	2	3	4	5	6	7	8	9	ZERO	HYPHEN	EQUAL
EH 1	EN 2	THIN 3	\$ 4	% 5	4 6	8 7	* 8	(9	) ZERO	1/4	Beta (=)
BACK SPACE											

Q	W	E	R	T	Y	U	I	O	P	EXCLAMATION POINT
Q 9	W 3	E 2	R 2	T 4	Y 4	U 3	I 2	O 0	P 0	degree

A	S	D	F	G	H	J	K	L	COLON	QUOTE
A 2	S 2	D 2	F 2	G 2	H 2	J 2	K 2	L 2	:	6 (open)

Z	X	C	V	B	N	M	COMMA	PERIOD	SHIELD	SHIFT
Z 2	X 2	C 2	V 2	B 2	N 2	M 2	,	.	?	

MAR REL	CLR	SET
---------	-----	-----

ATTN	OF
------	----

SPACE BAR

For Roman, Italic, bold, bold italic, serif or sans serif

Rho = 8 B
Delta = 8 C
Gamma = 8 F

IBM DATATEXT

B1	B2	B3	B4	B5	B6
Var. W/S	EN Dash				

FAMILY

D 2 (For Photronic CRT Output)

MAR REL	CLR	SET
---------	-----	-----

ONE	2	3	4	5	6	7	8	9	ZERO	HYPHEN	EQUAL	BACK SPACE	ATTN
EM 1	EN 2	THIN 3	lc 4	5	! 6	sc 7	sc 8	9		1/4	Beta (=)	BACK SPACE	ATTN
Q	Q	W	E	R	T	Y	U	I	O	P	EXCLAMATION POINT		
Q	W	E	R	T	Y	U	I	O	P	U.C.			
A	S	D	F	G	H	J	K	L	COLON	QUOTE		RETURN	
A	S	D	F	G	H	J	K	L	:	U.C.			
A	S	D	F	G	H	J	K	L	;	U.C.			
Z	X	C	V	B	N	M	COMMA	PERIOD	SHIELD				
Z	X	C	V	B	N	M	,	.		U.C.			
SHIFT										SHIFT			

SPACE BAR

Cap & Small cap - diphthongs (incl. lc), old style figures, id

For ROMAN ONLY

Rho = 8 B
Delta = 8 C
Gamma = 8 F

IBM DATATEXT

FAMILY

D3 (For Fototronics CRT Output)

B1	B2	B3	B4	B5	B6
Var. W/S	EN Dash				

MAR REL

CLR

SET

ONE	2	3	4	5	6	7	8	9	ZERO	HYPHEN	EQUAL
EM	EN	THIN									Beta (=)

Q	W	E	R	T	Y	U	I	O	P	EXCLAMATION POINT

A	S	D	F	G	H	J	K	L	COLON	QUOTE
U.C. /	U.C. /	U.C. /	U.C. /	U.C. /	U.C. /	U.C. /	U.C. /	U.C. /	U.C. /	U.C. /
l.c. /	l.c. /	l.c. /	l.c. /	l.c. /	l.c. /	l.c. /	l.c. /	l.c. /	l.c. /	l.c. /

Z	X	C	V	B	N	M	COMMA	PERIOD	SHIELD
S.C. /	S.C. /	S.C. /	S.C. /	S.C. /	S.C. /	S.C. /	S.C. /	S.C. /	S.C. /

TAB

LOCK

SHIFT

RETURN

BACK SPACE

ATTN

ON

OFF

SPACE BAR

Floating Accents

Can be used in Super 8 Transfer Positions

Rho = 8 B
Delta = 8 C
Gamma = 8 F

IBM DATATEXT

FAMILY d4 (for Photographic Output)

B1	B2	B3	B4	B5	B6
Var. w/s	EN Dash		✓	⋈	

MAR REL

CLR

SET

ONE		2	3	4	5	6	7	8	9	ZERO	HYPHEN	EQUAL
Q	W	E	R	T	Y	U	I	O	P	EXCLAMATION POINT	Beta (=)	
		E	P	I	W	⊕	I	O	π			
		ε	ρ	↖	ν	θ	ι	ο	↗			

LOCK		A	S	D	F	G	H	J	K	L	COLON	QUOTE
		A	W	Δ	⊗	I	H	≡	K	Δ	•	
		α	δ	8	ρ	λ	η	ε	κ	λ	5	

SHIFT		Z	X	C	V	B	N	M	COMMA	PERIOD	SHIELD	SHIFT	
		Z	X	ψ	Ω	B	N	M	,	.			
		s	x	φ	ω	β	γ	μ		⊙			

SPACE BAR

GREEK

RETURN

BACK SPACE

OFF

Can be used in Supercor & Inferior Positions

Rho = 8 B
Delta = 8 C
Gamma = 8 F

IBM DATATEXT

FAMILY

D6 (For Fototronic CRT Output)

B1	B2	B3	B4	B5	B6
Var. W/S	EN Dash		VR4	VR5	

MAR REL

ONE	2	3	4	5	6	7	8	9	ZERO	HYPHEN	EQUAL
EM	EN	THW	(A. q. d.)		VR6	VR8	VR9			1/4	Beta (=)
—	—	—	—	—	—	—	—	—	—	hph	

BACK SPACE

TAB	Q	W	E	R	T	Y	U	I	O	P	EXCLAMATION POINT
	@	SS (C)	P	SS (R)	+	8		≤	{	<	VR3
		(C)	R	(R)		9		≥	3	>	bld

LOCK	A	S	D	F	G	H	J	K	L	COLON	QUOTE
	Φ	*	↑	↑	□	—	—	△	7pt	2pt	"
			↓	↓	■	—	—	△	5pt	1pt	"

RETURN

CLR SET

SHIFT	Z	X	C	V	B	N	M	COMMA	PERIOD	SHIELD
	+	—	÷	+	#	<	>	CTP.	bld	VR1
			X		8	>	<	1pt	1pt	VR9

SHIFT

SPACE BAR

UNIVERSAL PECULIARS

3.2.27 CUTE

CUTE, for the "Cornell University Text Editor," is a program that can be used to add a professional quality to such documents as Manual texts, term-papers, and theses. It is versatile and yet very easy to use.

CUTE/360 - PROGRAMMER'S GUIDE

*The Cornell University Text Editor (CUTE)

Abstract

The Cornell University text editor is implemented to allow for:

- (a) texts whose line-width is 64 characters or less. (lines can be as short as one character.)
- (b) precise carriage control of edited output:
permits:-
 - (i) underscoring
 - (ii) choice of 1st print line on page
 - (iii) choice of last print line on page
 - (iv) line spacing
 - (v) print line for page number (bottom)
 - (vi) forced page ejection and line spacing
- (c) text-centering
- (d) right-margin justification
- (e) character-mode translation feature -- upper and lower case -- accepting mixed-mode input
- (f) line-packing and hence text-insertion capability
- (g) page control and page numbering in both manual and automatic modes
- (h) generation of input for a subsequent edit
- (i) easy-to-read and easy-to-prepare input text -- a type-written text can be channelled for keypunching with a minimum of editing
- (j) multiple editing of a line image
- (k) on-line storage and editing of input files (COOL facility)
- (l) absolute space-conservation in input text owing to use of very few control characters (a total of 3) which do not occupy extra space.

* Designed by E. C. Ogbuobiri and G. H. Blomgren

Programmed by G. H. Blomgren, Arlyn Kerr, and
E. C. Ogbuobiri

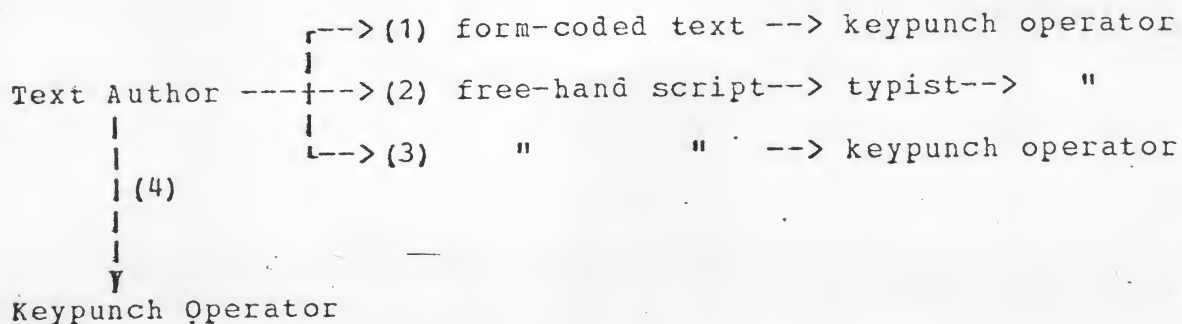
TABLE OF CONTENTS

	PAGE
Abstract	3-51
Table of Contents	3-52
Design Philosophy	3-53
CUTE Design Features	3-54
CUTE Design Limitations	3-56
Job Control Language	3-57
Invoking the CUTE Program	3-57
Text Description	3-60
CUTE Parameter Card	3-61
CUTE Text Cards	3-63
Programming Considerations	3-65
Appendix A :- Source Text For This Manual	3-67
Appendix B :- TN Characters and EBCDIC Punch-Card Codes	3-78

CUTE

Design Philosophy

Through a long history of collaboration, a manuscript writer has learned to communicate with a typist through a minimum of codes. Some of these codes are (a) line indenting for paragraph specification, (b) use of capital and lower case letters, etc. Nowadays, typing a manuscript is part of the training of a secretary. Of course a secretary may still prefer to type from a previously typed or draft copy which would invariably require that authors be typists themselves - a requirement which is against the principle of division of labor and skill allocation. For the keypunch operator, the relationship with authors is more scientific. Input text is not written free-hand, but is usually coded on special forms character by character, so that the keypunch operator is not burdened by artistic skills of recognition, interpretation, and re-arrangement, as a typist would often do. But coding a long text character by character on a special form is admittedly laborious and slow (much slower than a cursive script writing). The speed of getting materials ready for the keypunch operator can, however, be enhanced by interfacing the text author and the keypuncher with a typist. Greater speed can be achieved by encouraging the keypuncher to work directly from a manuscript cursively written. Thirdly, even greater speed can be achieved by having the author be the keypuncher. The following interface structure is thus described as channels (1), (2), (3), (4).



Channel (4) is obviously the fastest and corresponds to dictation (dictating while the keypuncher is punching). Channel (1) is invariably the slowest for large-volume texts.

CUTE is designed to facilitate the use of any of the four channels described above. Channel (1) is not new. Channel (2) is achieved through a type-coding form (form # OCSPTC) which was motivated by CUTE. Channel (3) is achieved through the use of coding standards which require minimum of skill on the part of a key-punch operator for direct execution: There are only three special control characters which (1) do not occupy extra space, and (2) can be activated by regular manuscript-coding standards familiar to all English readers and writers. There is also a text description which can be done by the author in minimum time and independently of the main text preparation. The description scheme is done in two ways: text requirements are described through a single parameter card while line characteristics are described on the margin. The use of Channel (4) depends on the competence of the user.

The above communication-oriented design philosophy of CUTE was subject to the following performance constraints:

- (1) versatility and flexibility
- (2) simplicity and readability of input text
- (3) fast execution time
- (4) small core requirements
- (5) text-handling capacity

Capacity must be limited only by the amount of auxiliary storage available. Simultaneous optimization in the light of constraints (1), (2), (3), and (4) leads to the implementation of only primitive text attributes in a modular form. Higher-level or less-frequently-used attributes which can however be programmed from the basic attributes are as far as possible excluded.

CUTE Design Features

- (A) CUTE assumes that a text is a string of lines whose attributes are left margin, right margin, character mode, printing position on a form (or carriage control).
- (B) Of all the attributes of a line, only the character mode is described within the text. The rest are either described on the margin or deferred until the printing of text is desired. The description of character mode does not increase the size of the actual text.

- (C) The input text is in EBCDIC. Cornell Users can also use BCD codes (026 keypunch codes) provided that they select the appropriate option on the COOL job card. If the text is punched in EBCDIC (029 keypunch codes), the character strings can be either in upper-case, lower-case, or a mixture of upper and lower-case characters.
- (D) CUTE is a complete processor requiring only the input text cards.
- (E) The author of the text provides editing information in two ways:
 - (a) through one parameter card describing the general features of the text described. Except for line-spacing, thesis-standard defaults are provided for a blank parameter card.
 - (b) Through editing information at the beginning and end of paragraphs only; a paragraph can be as short as one line. The paragraph-editing information requires a maximum of 5 entries corresponding to the five attributes of a line of text; all, some, or none of these attributes may be used.
- (F) Parameter-card information can be changed at any time through the passing of another parameter card. A batch processing feature is thus built into CUTE.
- (G) CUTE is modular in construction and thus allows new features to be simply incorporated. There are three programs:
 - (1) The Main Program TEXEDT (alias CUTE) scans the input and uses:
 - (2) Subroutine TEXOUT to deliver the output of edited text.
 - (3) Subroutine LINEDT to build a line, translate, center, and/or right-justify it.
- (H) CUTE always gives a printed output of edited text. A punched output of edited text can also be produced at the user's option.

CUTE Design Limitations

1. Maximum text-width is limited to 64 characters. This can easily be eliminated by providing continuation-card feature which at present is absent. There does not seem to be wide application for texts wider than 63 characters. (Thesis standard is 60 for PICA letter size which is the size used on line-printers.)
2. Words cannot be underscored until the author knows their exact position in the text. When CUTE is processing in the automatic line-packing mode, this information is not available until the first proofreading of the edited output. This limitation is tolerated in lieu of introducing another control character in the body of text.
3. There is no on-line input-file editing capability in CUTE. This capability is redundant in the COOL-HASP system, but might be desirable in other systems.
4. CUTE reasonably expects that the typist or the keypunch operator has a flexible control over the left margin alignment and hence has no tabulator feature. Note that this implies that leading blanks in a text field are significant. There is already a primitive which will allow inclusion of the tabulator feature and should this design assumption be wrong, tabulation can be included at little extra cost. In particular, right-justification of a single word within a text field, causes the single word to be moved to the end of the field.

JOB-CONTROL LANGUAGE

Invoking the CUTE Program:

Statement	b	Verb	b	Operand
Identification	l		l	
(starts col.1)	a		a	
	n		n	
	k		k	
//		EXEC		PGM=CUTE
//FT07F001		DD		route of punched output if any
//FT06F001		DD		route of printed output
//FT05F001		DD		location of input stream

- (a) route of punched output may be cards, tape, or disk.
- (b) route of printed output may be printer, tape, or disk.
- (c) location of input stream may be cards, tape, or disk.

A typical OS Job Step for which the text description is on cards is as follows:

```
//MYSTEP EXEC PGM=CUTE
//FT07F001 DD SYSOUT=B
//FT06F001 DD SYSOUT=A,DCB=(RECFM=UA,BLKSIZE=133)
//FT05F001 DD *
```

```
-----
| text description: 1st card must be a parameter |
|                  card and last card must be a  |
|                  CUTE end-of-file card          |
|-----|
```

/*

In the above example, any punched output is on cards and the printed output is via the line printer. The following job step is also equivalent but assumes that the print train is mounted on printer f :

```
//MYSTEP EXEC CUTE
//SYSIN DD *
```

```
-----
| text description: 1st card must be a parameter |
|                  card and last card must be a  |
|                  CUTE end-of-file card          |
|-----|
```

/*

In a COOL-HASP environment such as Cornell, a typical job deck for CUTE may take the form:

```
/JOB ...  
/EDIT  
//MYSTEP EXEC CUTE  
//SYSIN DD *  
/LIST
```

```
| text description: 1st card must be a parameter  
| card and last card must be a  
| CUTE end-of-file card  
|
```

```
/NOLIST  
/* THIS CARD IS OPTIONAL  
/OS bbbe  
/ENDJOB
```

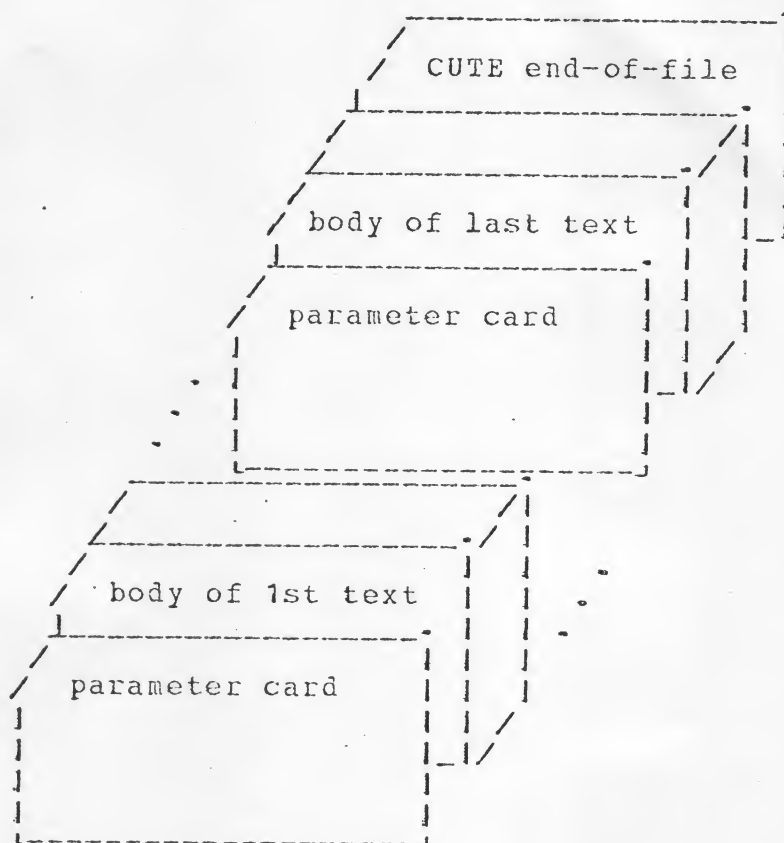
In the above COOL job deck:

1. The input card images will be listed
2. The edited text will be printed
3. Any punched output will be filed as a CDS named bbbe; if bbbe is blank, any punched output will be on cards.

TEXT DESCRIPTION

Text can be described on form OCSPTC which is also suitable for type-coding using a typewriter with pica-size letters.

text is described in the following sequence:



Apart from the first parameter card, subsequent parameter cards must be immediately preceded by a card whose column 8 has a three-punch; this card must be a text card.

CUTE PARAMETER CARD

The format is as follows:

Field	Card Column	Content
1	1-2	An integer number describing the ordinal position of the first printed line of every page; this number is right-justified in field and cannot be less than 05; if field is blank, 05 is assumed. (LINSTA)
2	4-5	An integer number denoting the ordinal position of the last printed line of text on each page; this number is right-justified in field and cannot be less than 05; if field is blank, 60 is assumed. (LINEND)
3	7	A number equal to the number of spaces between two adjacent lines of text; zero or blank denotes single spacing. (LINSPP)
4	9-10	An integer number describing the ordinal position of the line on which the page number is to be printed; this number is right-justified in field and must be greater than the content of field 2; the contents of field 2 (or its default value) increased by 3 is substituted in default. (LINPAN)
5	15-16	An integer number right-justified in the field. If the number is 00 or if the field is blank, automatic pagination is suppressed and page numbers are assumed to be supplied on page-eject cards. If the number is negative, page numbers are blanked out. If the number is 1, pagination is automatic, forced page-eject cards are also recognized, and page numbers are composed from fields 6, 7, and 8. (KPAGE)

Field	Card Column	Content
6	18-25	Page number prefix of 8 characters or less. trailing blanks are significant in this field. (PREFIX)
7	27-30	An integer number right-justified in the field. This number is the starting page number when page-numbering is automatic. A zero value or a blank field has the following interpretation: For the first parameter card in the job deck, it means that 4 blank spaces are inserted between the prefix and the suffix; for the subsequent parameter cards, it means that the current page of text is still in progress and that current page-number sequence should be continued. (INIPAG)
8	33-40	Page number suffix of 8 characters or less. Leading blanks are significant in this field. (SUFFIX)
9	51	Blank or zero punch indicates that no punched output of edited text is desired. A 1 punch indicates that a punched output of edited text is desired and that this punched output will be used as input to the next CUTE execution. When punching is desired, (1) the parameter card is punched with blank fields 9, 10, and 11, (2) all subsequent cards are punched except cards used for multiple editing of a line image, (3) each card punched contains carriage control information in columns 1-2, and a 3 or 4 punch in column 8 as desired, the line of text, and the serial number of the card in columns 73-80. (KPUNCH)
10	55-60	An integer number denoting the starting serial number of punched cards. This number must be right-justified in field. COOL file sequence numbers override this if punched output is filed as a CDS. (KPSER)
11	62-80	Optional comments

CUTE TEXT CARDS

(All integer entries in fields 1, 3, 4 must be right justified.)

Field	Card Col.	Punch	Meaning
1	1-2	blank 01 02 03 N≥10	carriage control as on parameter card. suppress space before printing edited line of text end-of-file eject page; if pagination is in manual mode, current page number is taken from this card, otherwise not. skip the next N-10 lines before printing this line of text.
2	3	blank 1 2 3	no editing is required on line of text. translate only (using fields 3 and 4) translate and center between left and right margin as defined in fields 3 and 4. translate and right-margin justify at right margin as defined in field 4.
3	4-5	N1	base-one address of left margin byte of text. $1 \leq N1 \leq N2 \leq 64$. $N1 = (\text{card col.} - 8)$.
4	6-7	N2	base-one address of right-margin byte of text. $1 \leq N1 \leq N2 \leq 64$. $N2 = (\text{card column} - 8)$.
5	8	blank 1 2 3 4 5 6	no information insertion with line-packing begins with this card. The values of N1 and N2 given on this card are repeatedly used until the packing mode is turned off. insertion with line-packing ends with this card. next card in sequence is a parameter card. this card is a comment card. this line will be multiply edited using fields 1, 2, 3, 4, 6 of this card and fields 2, 3, 4 of following cards. multiple editing of current line terminates with this card.

Field	Card Col.	Punch	Meaning
6	9-72	TEXT	It is recommended for thesis standard that text end in column 69 or 70. Column 9 of card must either be blank or contain one of the three shift characters described below. During an operation involving centering, packing, or a right-justification, the leading blanks of a word of text are significant but the trailing blanks of any line of text are not; in this circumstance, the shift characters are treated as blanks. During translation, the shift characters are replaced by EBCDIC codes which do not have printing graphics. In this way alphabetic case information is preserved for later passes through the editor of punched output, but they are blanked for printing. The shift characters are used as follows: ⓪ replace by a blank and capitalize the first letter of immediate word. Ⓢ replace by a blank and capitalize the immediate word. Ⓜ replace by a blank and capitalize the rest of the line of text. The user is free to use some contiguous subset of this field (or several subsets in the multiple-edit mode). The program does not, however, destroy (blank out) any information punched in text columns beyond the user-indicated bounds. The entire field is printed out after edit.
7	73-80		optional comments

Programming Considerations

1. CUTE input text may be manually or type-coded on form OCSPTC available through the bookstore.
2. To minimize the editing time spent on input text preparation it is recommended that texts be processed in packing mode whenever practicable. In this mode, CUTE can pack a paragraph using information provided on the first and last cards of the paragraph. The first card provides the line width and subsequently, lines are packed, translated and right-justified until packing is turned off by the last card of the paragraph. In packing mode, page-eject and underscore cards are recognized and properly buffered. Centering or translate-only operations are not recognized, nor does the program heed any changes in N1 or N2. The user is therefore cautioned to be sure to turn off the packing mode at the end of each paragraph.
3. A character in column 10 of input card is printed in print position 21 of the line printer. This information will be useful in aligning the edge of special output forms.
4. Texts for multiple reproduction should be printed onto a magnetic tape which can be dumped repeatedly by a tape-to-print processor.
5. Error messages are returned in place of the text lines that trigger them. The message is preceded and followed by stars.
6. Core requirement for the program is less than 21K bytes.
7. The first parameter card always forces a page-eject. Subsequent parameter cards do not necessarily force a page-eject.

8. A line of text, as is printed out, consists of any characters that are contained in columns 9 through 72 of the input card.
9. The trailing blanks of each line of text are not significant.
10. During the right-margin justification of a group of words within a specified field, the first word within the field is not displaced unless the field contains only one word whereupon the word is moved to the end of the field. This fact, together with the multiple-edit feature can be used to move a group of words within a line of text. The user should note that the alphabetic-shift-control characters @ and ¢ must be attached to the words they modify; the control character ~ must be contiguous with the first of the group of words it modifies.
11. Only integer numbers may appear in columns 1 through 8 of any input card. Failure to observe this rule will terminate the job with the system I/O error message No. IHC215I. A parameter card, other than the very first one, must be immediately preceded by a card which contains a three punch in column 8.

SOURCE TEXT FOR THIS MANUAL

PARAMETER CARD

```

7          1          3-  51
00201620-CUTE/360 - PROGRAMMER'S GUIDE
13201620@*THE@CORNELL@UNIVERSITY@TEXT@EDITOR@ (CUTE)
1310162 @ABSTRACT
01
1330162 @THE@CORNELL@UNIVERSITY TEXT EDITOR IS IMPLEMENTED TO ALLOW
0010162 FOR:
00311625 (A) TEXTS WHOSE LINE-WIDTH IS 64 CHARACTERS OR LESS.
108086
0010162 (LINES CAN BE AS SHORT AS ONE CHARACTER.)
0010162 (B) PRECISE CARRIAGE CONTROL OF EDITED OUTPUT:
0010162 PERMITS:- (I) UNDERSCORING
00101625 (II) CHOICE OF 1ST PRINT LINE ON PAGE
00327626 (III) CHOICE OF LAST PRINT LINE ON PAGE
00101625 (IV) LINE SPACING
00327626 (V) PRINT LINE FOR PAGE NUMBER (BOTTOM)
0010162 (VI) FORCED PAGE EJECTION AND LINE
0010162 SPACING
0010162 (C) TEXT-CENTERING
0010162 (D) RIGHT-MARGIN JUSTIFICATION
0010162 (E) CHARACTER-MODE TRANSLATION FEATURE -- UPPER AND LOWER
0010162 CASE -- ACCEPTING MIXED-MODE INPUT
00101625 (F) LINE-PACKING AND HENCE TEXT-INSERTION CAPABILITY
00311626
00101625 (G) PAGE CONTROL AND PAGE NUMBERING IN BOTH MANUAL AND
00311626 AUTOMATIC MODES
0010162 (H) GENERATION OF INPUT FOR A SUBSEQUENT EDIT
00101625 (I) EASY-TO-READ AND EASY-TO-PREPARE INPUT TEXT -- A
00311626 TYPE-WRITTEN TEXT CAN BE CHANNELLED FOR KEYPUNCHING
0010162 WITH A MINIMUM OF EDITING
0010162 (J) MULTIPLE EDITING OF A LINE IMAGE
00101625 (K) ON-LINE STORAGE AND EDITING OF INPUT FILES@ (COOL
00311626 FACILITY)
0010162 (L) ABSOLUTE SPACE-CONSERVATION IN INPUT TEXT OWING TO
00101625 USE OF VERY FEW CONTROL CHARACTERS (A TOTAL OF 3)
00311626 WHICH DO NOT OCCUPY EXTRA SPACE.
0010162
13
1110162 *@DESIGNED BY @E.@C.@OGBUOBIRI AND @G.@H.@BLOMGREN
01
1110162 @PROGRAMMED BY @G.@H.@BLOMGREN,@ARLYN@KERR, AND
01
0010162 @E.@C.@OGBUOBIRI
03
20162 -TABLE OF CONTENTS
1330162 @PAGE
1310162 @ABSTRACT
1210162 @TABLE OF@CONTENTS

```

3-51

3-52

1210162	@DESIGN@PHILOSOPHY	3-53
1210162	¿CUTE@DESIGN@FEATURES	3-54
1210162	¿CUTE@DESIGN@LIMITATIONS	3-56
1210162	@JOB@CONTROL@LANGUAGE	3-57
1110162	@INVOKING THE¿CUTE@PROGRAM	3-57
1110162	@TEXT@DESCRIPTION	3-60
1110162	¿CUTE@PARAMETER@CARD	3-61
1110162	¿CUTE@TEXT@CARDS	3-63
1210162	@PROGRAMMING@CONSIDERATIONS	3-65
1210162	@APPENDIX@A :- @SOURCE@TEXT@FOR@THIS@MANUAL	
1210162	@APPENDIX@B :- ¿TN@CHARACTERS AND¿EBCDIC@PUNCH@CARD@CODES	

01

03

2 162 -CUTE

131 162 @DESIGN@PHILOSOPHY .

1

133 1621@THROUGH A LONG HISTORY OF COLLABORATION, A MANUSCRIPT WRITER HAS LEARNED TO COMMUNICATE WITH A TYPIST THROUGH A MINIMUM OF CODES. @SOME OF THESE CODES ARE (A) LINE INDENTING FOR PARAGRAPH SPECIFICATION, (B) USE OF CAPITAL AND LOWER CASE LETTERS, ETC. @NOWADAYS, TYPING A MANUSCRIPT IS PART OF THE TRAINING OF A SECRETARY. @OF COURSE A SECRETARY MAY STILL PREFER TO TYPE FROM A PREVIOUSLY TYPED OR DRAFT COPY WHICH WOULD INVARIABLY REQUIRE THAT AUTHORS BE TYPISTS THEMSELVES - A REQUIREMENT WHICH IS AGAINST THE PRICIPLE OF DIVISION OF LABOR AND SKILL ALLOCATION. @FOR THE KEYPUNCH OPERATOR, THE RELATIONSHIP WITH AUTHORS IS MORE SCIENTIFIC. @INPUT TEXT IS NOT WRITTEN FREE-HAND, BUT IS USUALLY CODED ON SPECIAL FORMS CHARACTER BY CHARACTER, SO THAT THE KEYPUNCH OPERATOR IS NOT BURDENED BY ARTISTIC SKILLS OF RECOGNITION, INTERPRETATION, AND RE-ARRANGEMENT, AS A TYPIST WOULD OFTEN DO. @BUT CODING A LONG TEXT CHARACTER BY CHARACTER ON A SPECIAL FORM IS ADMITTEDLY LABORIOUS AND SLOW (MUCH SLOWER THAN A CURSIVE SCRIPT WRITING). @THE SPEED OF GETTING MATERIALS READY FOR THE KEYPUNCH OPERATOR CAN, HOWEVER, BE ENHANCED BY INTERFACING THE TEXT AUTHOR AND THE KEYPUNCHER WITH A TYPIST. @GREATER SPEED CAN BE ACHIEVED BY ENCOURAGING THE KEYPUNCHER TO WORK DIRECTLY FROM A MANUSCRIPT CURSIVELY WRITTEN. @THIRDLY, EVEN GREATER SPEED CAN BE ACHIEVED BY HAVING THE AUTHOR BE THE KEYPUNCHER. @THE FOLLOWING INTERFACE STRUCTURE IS THUS DESCRIBED AS

2 CHANNELS (1), (2), (3), (4) -

13101620		---	(1)	FORM-CODED TEXT	-->	KEYPUNCH OPERATOR
00101620						
00101620	@TEXT@AUTHOR	---		-->	(2)	FREE-HAND SCRIPT--> TYPIST--> "
01						
00101620						
00101620				L-->	(3)	" " --> KEYPUNCH OPERATOR
00101620			(4)			
00101620						
00101620						
00101620						
01		V				
00101620	@KEYPUNCH@OPERATOR					

133 1621@CHANNEL (4) IS OBVIOUSLY THE FASTEST AND CORRESPONDS TO
 DICTATION (DICTATING WHILE THE KEYPUNCHER IS PUNCHING).
 2@CHANNEL (1) IS INVARIABLY THE SLOWEST FOR LARGE-VOLUME TEXTS.

133 1621@CUTE IS DESIGNED TO FACILITATE THE USE OF ANY OF THE FOUR
 CHANNELS DESCRIBED ABOVE. @CHANNEL (1) IS NOT NEW. @CHANNEL
 (2) IS ACHIEVED THROUGH A TYPE-CODING FORM (FORM #OCSPTC)
 WHICH WAS MOTIVATED BY@CUTE. @CHANNEL (3) IS ACHIEVED THROUGH
 THE USE OF CODING STANDARDS WHICH REQUIRE MINIMUM OF SKILL ON
 THE PART OF A KEY-PUNCH OPERATOR FOR DIRECT EXECUTION: @THERE
 ARE ONLY THREE SPECIAL CONTROL CHARACTERS WHICH (1) DO NOT
 OCCUPY EXTRA SPACE, AND (2) CAN BE ACTIVATED BY REGULAR
 MANUSCRIPT-CODING STANDARDS FAMILIAR TO ALL@ENGLISH READERS
 AND WRITERS. @THERE IS ALSO A TEXT DESCRIPTION WHICH CAN BE
 DONE BY THE AUTHOR IN MINIMUM TIME AND INDEPENDENTLY OF THE
 MAIN TEXT PREPARATION. @THE DESCRIPTION SCHEME IS DONE IN TWO
 WAYS: TEXT REQUIREMENTS ARE DESCRIBED THROUGH A SINGLE
 PARAMETER CARD WHILE LINE CHARACTERISTICS ARE DESCRIBED ON
 THE MARGIN. @THE USE OF@CHANNEL (4) DEPENDS ON THE COMPETENCE
 2 OF THE USER.

13301621@THE ABOVE COMMUNICATION-ORIENTED DESIGN PHILOSOPHY OF@CUTE
 2 WAS SUBJECT TO THE FOLLOWING PERFORMANCE CONSTRAINTS:

1110162 (1) VERSATILITY AND FLEXIBILITY
 10162 (2) SIMPLICITY AND READABILITY OF INPUT TEXT
 10162 (3) FAST EXECUTION TIME
 10162 (4) SMALL CORE REQUIREMENTS
 10162 (5) TEXT-HANDLING CAPACITY

13301621@CAPACITY MUST BE LIMITED ONLY BY THE AMOUNT OF AUXILIARY
 STORAGE AVAILABLE. @SIMULTANEOUS OPTIMIZATION IN THE LIGHT OF
 CONSTRAINTS (1), (2), (3), AND (4) LEADS TO THE
 IMPLEMENTATION OF ONLY PRIMITIVE TEXT ATTRIBUTES IN A MODULAR
 FORM. @HIGHER-LEVEL OR LESS-FREQUENTLY-USED ATTRIBUTES WHICH
 CAN HOWEVER BE PROGRAMMED FROM THE BASIC ATTRIBUTES ARE AS
 2 FAR AS POSSIBLE EXCLUDED.

1310162 @CUTE@DESIGN@FEATURES
 01

13306621 (A) @CUTE ASSUMES THAT A TEXT IS A STRING OF LINES WHOSE
 2 ATTRIBUTES ARE LEFT MARGIN, RIGHT MARGIN, CHARACTER
 MODE, PRINTING POSITION ON A FORM (OR CARRIAGE CONTROL).

1230662 (B) @OF ALL THE ATTRIBUTES OF A LINE, ONLY THE CHARACTER MODE
 306621 IS DESCRIBED WITHIN THE TEXT. @THE REST ARE EITHER
 DESCRIBED ON THE MARGIN OR DEFERRED UNTIL THE PRINTING
 2 OF TEXT IS DESIRED. @THE DESCRIPTION OF CHARACTER MODE
 DOES NOT INCREASE THE SIZE OF THE ACTUAL TEXT.

12306621 (C) @THE INPUT TEXT IS IN@EBCDIC. @CORNELL@USERS CAN ALSO USE
 @BCD CODES (026 KEYPUNCH CODES) PROVIDED THAT THEY SELECT
 THE APPROPRIATE OPTION ON THE@COOL JOB CARD. @IF THE
 2 TEXT IS PUNCHED IN@EBCDIC (029 KEYPUNCH CODES), THE
 CHARACTER STRINGS CAN BE EITHER IN UPPER-CASE,
 LOWER-CASE, OR A MIXTURE OF UPPER AND LOWER-CASE
 2 CHARACTERS.

12306621 (D) @CUTE IS A COMPLETE PROCESSOR REQUIRING ONLY THE INPUT
 2 TEXT CARDS.

12306621 (E) @THE AUTHOR OF THE TEXT PROVIDES EDITING INFORMATION IN
 2 TWO WAYS:

11311625 (A) THROUGH ONE PARAMETER CARD DESCRIBING THE GENERAL
108086 FEATURES OF THE TEXT DESCRIBED. @EXCEPT FOR
311621 LINE-SPACING, THESIS-STANDARD DEFAULTS ARE PROVIDED
2 FOR A BLANK PARAMETER CARD.

11311625 (B) @THROUGH EDITING INFORMATION AT THE BEGINNING AND
108086
311621 END OF PARAGRAPHS ONLY; A PARAGRAPH CAN BE AS SHORT
AS ONE LINE. @THE PARAGRAPH-EDITING INFORMATION
REQUIRES A MAXIMUM OF 5 ENTRIES CORRESPONDING TO
THE FIVE ATTRIBUTES OF A LINE OF TEXT; ALL, SOME,
OR NONE OF THESE ATTRIBUTES MAY BE USED.

12306621 (F) @PARAMETER-CARD INFORMATION CAN BE CHANGED AT ANY TIME
2 THROUGH THE PASSING OF ANOTHER PARAMETER CARD. @A BATCH
2 PROCESSING FEATURE IS THUS BUILT INTO@CUTE.

12306621 (G) @CUTE IS MODULAR IN CONSTRUCTION AND THUS ALLOWS NEW
2 FEATURES TO BE SIMPLY INCORPORATED. @THERE ARE THREE
PROGRAMS:
11311621 (1) @THE@MAIN@PROGRAM@TEXEDT (ALIAS@CUTE) SCANS THE
2 INPUT AND USES:
11311621 (2) @SUBROUTINE@TEXOUT TO DELIVER THE OUTPUT OF EDITED
2 TEXT.
11311621 (3) @SUBROUTINE@LINEDT TO BUILD A LINE, TRANSLATE,
2 CENTER, AND/OR RIGHT-JUSTIFY IT.

12306621 (H) @CUTE ALWAYS GIVES A PRINTED OUTPUT OF EDITED TEXT. @A
2 PUNCHED OUTPUT OF EDITED TEXT CAN ALSO BE PRODUCED AT
THE USER'S OPTION.

03
10162 @CUTE@DESIGN@LIMITATIONS

01
0

12306621 1. @MAXIMUM TEXT-WIDTH IS LIMITED TO 64 CHARACTERS. @THIS
CAN EASILY BE ELIMINATED BY PROVIDING CONTINUATION-CARD
FEATURE WHICH AT PRESENT IS ABSENT. @THERE DOES NOT SEEM
TO BE WIDE APPLICATION FOR TEXTS WIDER THAN 63
CHARACTERS. @ (THEESIS STANDARD IS 60 FOR@PICA LETTER SIZE
WHICH IS THE SIZE USED ON LINE-PRINTERS.)

12306621 2. @WORDS CANNOT BE UNDERScoreD UNTIL THE AUTHOR KNOWS THEIR
EXACT POSITION IN THE TEXT. @WHEN@CUTE IS PROCESSING IN
THE AUTOMATIC LINE-PACKING MODE, THIS INFORMATION IS NOT
AVAILABLE UNTIL THE FIRST PROOFREADING OF THE EDITED
OUTPUT. @THIS LIMITATION IS TOLERATED IN LIEU OF
INTRODUCING ANOTHER CONTROL CHARACTER IN THE BODY OF
TEXT.

12306621 3. @THERE IS NO ON-LINE INPUT-FILE EDITING CAPABILITY IN
@CUTE. @THIS CAPABILITY IS REDUNDANT IN THE@COOL-HASP
SYSTEM, BUT MIGHT BE DESIRABLE IN OTHER SYSTEMS.

12306621 4. @CUTE REASONABLY EXPECTS THAT THE TYPIST OR THE KEYPUNCH
OPERATOR HAS A FLEXIBLE CONTROL OVER THE LEFT MARGIN
ALIGNMENT AND HENCE HAS NO TABULATOR FEATURE. @NOTE THAT
THIS IMPLIES THAT LEADING BLANKS IN A TEXT FIELD ARE
SIGNIFICANT. @THERE IS ALREADY A PRIMITIVE WHICH WILL
ALLOW INCLUSION OF THE TABULATOR FEATURE AND SHOULD THIS
DESIGN ASSUMPTION BE WRONG, TABULATION CAN BE INCLUDED
AT LITTLE EXTRA COST. @IN PARTICULAR, RIGHT-JUSTIFICA-

2 TION OF A SINGLE WORD WITHIN A TEXT FIELD, CAUSES THE
SINGLE WORD TO BE MOVED TO THE END OF THE FIELD.

03

20162 -JOB-CONTROL LANGUAGE

0120162 -

1310162 @INVOKING THE@CUTE@PROGRAM:

01

13

01

10162	@STATEMENT	B @VERB	B @OPERAND
10162	@IDENTIFICATION	L	L
10162	(STARTS COL.1)	A	A
10162		N	N
10162		K	K

01

//		EXEC		PGM=CUTE

01

10162	@//FT07F001		@DD		ROUTE OF PUNCHED OUTPUT IF ANY

01

10162	@//FT06F001		@DD		ROUTE OF PRINTED OUTPUT

01

10162	@//FT05F001		@DD		LOCATION OF INPUT STREAM

01

13306625 (A) ROUTE OF PUNCHED OUTPUT MAY BE CARDS, TAPE, OR DISK.

103036

13306625 (B) ROUTE OF PRINTED OUTPUT MAY BE PRINTER, TAPE, OR DISK.

103036

13306625 (C) LOCATION OF INPUT STREAM MAY BE CARDS, TAPE, OR DISK.

103036

03

301621@A TYPICAL@OS@JOB@STEP FOR WHICH THE TEXT DESCRIPTION IS ON
2 CARDS IS AS FOLLOWS:

13

```
//MYSTEP EXEC PGM=CUTE
//FT07F001 DD SYSOUT=B
//FT06F001 DD SYSOUT=A,DCB=(RECFM=UA,BLKSIZE=133)
//FT05F001 DD *
```

10162	TEXT DESCRIPTION: 1ST CARD MUST BE A PARAMETER
10162	CARD AND LAST CARD MUST BE A
10162	@CUTE END-OF-FILE CARD

/*

13301621@IN THE ABOVE EXAMPLE, ANY PUNCHED OUTPUT IS ON CARDS AND THE
PRINTED OUTPUT IS VIA THE LINE PRINTER. @THE FOLLOWING JOB
STEP IS ALSO EQUIVALENT BUT ASSUMES THAT THE PRINT TRAIN IS

2 MOUNTED ON PRINTER F :

13

```
//MYSTEP EXEC CUTE
```

```
//SYSIN      DD  *
```

```
10162 | TEXT DESCRIPTION: 1ST CARD MUST BE A PARAMETER
10162 | CARD AND LAST CARD MUST BE A
10162 | CUTE END-OF-FILE CARD
```

```
/*
```

03
12301621 @IN A CUTE HASP ENVIRONMENT SUCH AS CORNELL, A TYPICAL
2 JOB DECK FOR CUTE MAY TAKE THE FORM:

```
13  /JOB ...
    /EDIT
    //MYSTEP EXEC CUTE
    //SYSIN   DD  *
    /LIST
```

```
10162 | TEXT DESCRIPTION: 1ST CARD MUST BE A PARAMETER
10162 | CARD AND LAST CARD MUST BE A
10162 | CUTE END-OF-FILE CARD
```

```
/NOLIST
```

```
/*
```

THIS CARD IS OPTIONAL

```
10162 /OS  BBBE
      /ENDJOB
```

1310162 @IN THE ABOVE CUTE JOB DECK:

- 1110162 1. @THE INPUT CARD IMAGES WILL BE LISTED
- 10162 2. @THE EDITED TEXT WILL BE PRINTED
- 311621 3. @ANY PUNCHED OUTPUT WILL BE FILED AS A CDS NAMED
BBBE; IF BBBE IS BLANK, ANY PUNCHED OUTPUT WILL BE
2 ON CARDS.

03

TEXT DESCRIPTION

01

13301621 @TEXT CAN BE DESCRIBED ON FORM OCSPTC WHICH IS ALSO SUITABLE
2 FOR TYPE-CODING USING A TYPEWRITER WITH PICA-SIZE LETTERS.

1310162 TEXT IS DESCRIBED IN THE FOLLOWING SEQUENCE:

13

10162

```
      / CUTE END-OF-FILE
```

01

01

10162

```
      / BODY OF LAST TEXT
```

01

10162

```
      / PARAMETER CARD
```

01

01

10162

BODY OF 1ST TEXT

01

10162

PARAMETER CARD

13301621@APART FROM THE FIRST PARAMETER CARD, SUBSEQUENT PARAMETER CARDS MUST BE IMMEDIATELY PRECEDED BY A CARD WHOSE COLUMN 8 2 HAS A THREE-PUNCH; THIS CARD MUST BE A TEXT CARD.

03

CUTE PARAMETER CARD

01

1310162 @THE FORMAT IS AS FOLLOWS:

1310162 @FIELD @CARD @CONTENT

10162 @COLUMN

12316621	1	1-2	@AN INTEGER NUMBER DESCRIBING THE ORDINAL POSITION OF THE FIRST PRINTED LINE OF EVERY PAGE; THIS NUMBER IS RIGHT-JUSTIFIED IN FIELD AND CANNOT BE LESS THAN 05; IF FIELD IS BLANK, 05 IS ASSUMED. ⑈(LINSTA)
12316621	2	4-5	@AN INTEGER NUMBER DENOTING THE ORDINAL POSITION OF THE LAST PRINTED LINE OF TEXT ON EACH PAGE; THIS NUMBER IS RIGHT-JUSTIFIED IN FIELD AND CANNOT BE LESS THAN 05; IF FIELD IS BLANK, 60 IS ASSUMED. ⑈(LINEND)
12316621	3	7	@A NUMBER EQUAL TO THE NUMBER OF SPACES BETWEEN TWO ADJACENT LINES OF TEXT; ZERO OR BLANK DENOTES SINGLE SPACING. ⑈(LINSIPA)
12316621	4	9-10	@AN INTEGER NUMBER DESCRIBING THE ORDINAL POSITION OF THE LINE ON WHICH THE PAGE NUMBER IS TO BE PRINTED; THIS NUMBER IS RIGHT-JUSTIFIED IN FIELD AND MUST BE GREATER THAN THE CONTENT OF FIELD 2; THE CONTENTS OF FIELD 2 (OR ITS DEFAULT VALUE) INCREASED BY 3 IS SUBSTITUTED IN DEFAULT. ⑈(LINPAN)
12316621	5	15-16	@AN INTEGER NUMBER RIGHT-JUSTIFIED IN THE FIELD. @IF THE NUMBER IS 00 OR IF THE FIELD IS BLANK, AUTOMATIC PAGINATION IS SUPPRESSED AND PAGE NUMBERS ARE ASSUMED TO BE SUPPLIED ON

PAGE-EJECT CARDS. @IF THE NUMBER IS NEGATIVE,
PAGE NUMBERS ARE BLANKED OUT. @IF THE NUMBER
IS 1, PAGINATION IS AUTOMATIC, FORCED
PAGE-EJECT CARDS ARE ALSO RECOGNIZED, AND PAGE
NUMBERS ARE COMPOSED FROM FIELDS 6, 7, AND 8.
⚡(KPAGE)

03	2			
		10162	@FIELD @CARD	@CONTENT
		10162	@COLUMN	
12316621	6	18-25	@PAGE NUMBER PREFIX OF 8 CHARACTERS OR LESS.	
			TRAILING BLANKS ARE SIGNIFICANT IN THIS FIELD.	
	2		⚡(PREFIX)	
12316621	7	27-30	@AN INTEGER NUMBER RIGHT-JUSTIFIED IN THE	
			FIELD. @THIS NUMBER IS THE STARTING PAGE	
			NUMBER WHEN PAGE-NUMBERING IS AUTOMATIC. @A	
			ZERO VALUE OR A BLANK FIELD HAS THE FOLLOWING	
			INTERPRETATION: @FOR THE FIRST PARAMETER CARD	
			IN THE JOB DECK, IT MEANS THAT 4 BLANK SPACES	
			ARE INSERTED BETWEEN THE PREFIX AND THE	
			SUFFIX; FOR THE SUBSEQUENT PARAMETER CARDS, IT	
			MEANS THAT THE CURRENT PAGE OF TEXT IS STILL	
			IN PROGRESS AND THAT CURRENT PAGE-NUMBER	
			SEQUENCE SHOULD BE CONTINUED. ⚡(INIPAG)	
12316621	8	33-40	@PAGE NUMBER SUFFIX OF 8 CHARACTERS OR LESS.	
			@LEADING BLANKS ARE SIGNIFICANT IN THIS FIELD.	
	2		⚡(SUFFIX)	
12316621	9	51	@BLANK OR ZERO PUNCH INDICATES THAT NO PUNCHED	
			OUTPUT OF EDITED TEXT IS DESIRED. @A 1 PUNCH	
			INDICATES THAT A PUNCHED OUTPUT OF EDITED TEXT	
			IS DESIRED AND THAT THIS PUNCHED OUTPUT WILL	
			BE USED AS INPUT TO THE NEXT CUTE EXECUTION.	
			@WHEN PUNCHING IS DESIRED, (1) THE PARAMETER	
			CARD IS PUNCHED WITH BLANK FIELDS 9, 10, AND	
			11, (2) ALL SUBSEQUENT CARDS ARE PUNCHED	
			EXCEPT CARDS USED FOR MULTIPLE EDITING OF A	
			LINE IMAGE, (3) EACH CARD PUNCHED CONTAINS	
			CARRIAGE CONTROL INFORMATION IN COLUMNS 1-2,	
			AND A 3 OR 4 PUNCH IN COLUMN 8 AS DESIRED, THE	
			LINE OF TEXT, AND THE SERIAL NUMBER OF THE	
			CARD IN COLUMNS 73-80. ⚡(KPUNCH)	
12316621	10	55-60	@AN INTEGER NUMBER DENOTING THE STARTING SERIAL	
			NUMBER OF PUNCHED CARDS. @THIS NUMBER MUST BE	
			RIGHT-JUSTIFIED IN FIELD. ⚡COOL FILE SEQUENCE	
			NUMBERS OVERRIDE THIS IF PUNCHED OUTPUT IS	
			FILED AS A CDS. ⚡(KPSER)	
1211662	11	62-80	@OPTIONAL COMMENTS	

03

CUTE TEXT CARDS

01

11301621	@	(ALL INTEGER ENTRIES IN FIELDS 1, 3, 4 MUST BE RIGHT
	2	JUSTIFIED.)
1210162	@FIELD @CARD	@PUNCH @MEANING
10162	@COL.	
12323625	1	1-2 BLANK CARRIAGE CONTROL AS ON PARAMETER CARD.

117216			01	SUPPRESS SPACE BEFORE PRINTING EDITED
32362				LINE OF TEXT
12362			02	END-OF-FILE
12362			03	EJECT PAGE; IF PAGINATION IS IN MANUAL
323621				MODE, CURRENT PAGE NUMBER IS TAKEN FROM
	2			THIS CARD, OTHERWISE NOT.
323621			N≥10	SKIP THE NEXT@N-10 LINES BEFORE
	2			PRINTING THIS LINE OF TEXT.
12323625	2	3	BLANK	NO EDITING IS REQUIRED ON LINE OF TEXT.
117216				
12362			1	TRANSLATE ONLY (USING FIELDS 3 AND 4)
323621			2	TRANSLATE AND CENTER BETWEEN LEFT AND
				RIGHT MARGIN AS DEFINED IN FIELDS 3
	2			AND 4.
323621			3	TRANSLATE AND RIGHT-MARGIN JUSTIFY AT
	2			RIGHT MARGIN AS DEFINED IN FIELD 4.
12323621	3	4-5	N1	BASE-ONE ADDRESS OF LEFT MARGIN BYTE OF
	2			TEXT. @1≤N1≤N2≤64. @N1 = (CARD COL.-8).
12323621	4	6-7	N2	BASE-ONE ADDRESS OF RIGHT-MARGIN BYTE
				OF TEXT. @1≤N1≤N2≤64. @N2 = (CARD
	2			COLUMN - 8).
1211762	5	8	BLANK	NO INFORMATION
323621			1	INSERTION WITH LINE-PACKING BEGINS WITH
				THIS CARD. @THE VALUES OF@N1 AND@N2
	2			GIVEN ON THIS CARD ARE REPEATEDLY USED
32362			2	UNTIL THE PACKING MODE IS TURNED OFF.
12362				INSERTION WITH LINE-PACKING ENDS WITH
32362			3	THIS CARD.
12362				NEXT CARD IN SEQUENCE IS A PARAMETER
12362			4	CARD.
12362			5	THIS CARD IS A COMMENT CARD.
32362				THIS LINE WILL BE MULTIPLY EDITED USING
12362				FIELDS 1, 2, 3, 4, 6 OF THIS CARD AND
323621				FIELDS 2, 3, 4 OF FOLLOWING CARDS.
	2		6	MULTIPLE EDITING OF CURRENT LINE
				TERMINATES WITH THIS CARD.
03				
10162	@FIELD	@CARD	@PUNCH	@MEANING
10162		@COL.		
12323621	6	9-72	TEXT	@IT IS RECOMMENDED FOR THESIS STANDARD
				THAT TEXT END IN COLUMN 69 OR 70.
				@COLUMN 9 OF CARD MUST EITHER BE BLANK
				OR CONTAIN ONE OF THE THREE SHIFT
				CHARACTERS DESCRIBED BELOW. @DURING AN
				OPERATION INVOLVING CENTERING, PACKING,
				OR A RIGHT-JUSTIFICATION, THE LEADING
				BLANKS OF A WORD OF TEXT ARE
				SIGNIFICANT BUT THE TRAILING BLANKS OF
				ANY LINE OF TEXT ARE NOT; IN THIS
				CIRCUMSTANCE, THE SHIFT CHARACTERS ARE
				TREATED AS BLANKS. @DURING TRANSLATION,
				THE SHIFT CHARACTERS ARE REPLACED BY
				@EBCDIC CODES WHICH DO NOT HAVE PRINTING

GRAPHICS. @IN THIS WAY ALPHABETIC CASE INFORMATION IS PRESERVED FOR LATER PASSES THROUGH THE EDITOR OF PUNCHED OUTPUT, BUT THEY ARE BLANKED FOR PRINTING. @THE SHIFT CHARACTERS ARE USED AS FOLLOWS:

- @ REPLACE BY A BLANK AND CAPITALIZE THE FIRST LETTER OF IMMEDIATE WORD.
- ¢ REPLACE BY A BLANK AND CAPITALIZE THE IMMEDIATE WORD.
- REPLACE BY A BLANK AND CAPITALIZE THE REST OF THE LINE OF TEXT.

@THE USER IS FREE TO USE SOME CONTIGUOUS SUBSET OF THIS FIELD (OR SEVERAL SUBSETS IN THE MULTIPLE-EDIT MODE). @THE PROGRAM DOES NOT, HOWEVER, DESTROY (BLANK OUT) ANY INFORMATION PUNCHED IN TEXT COLUMNS BEYOND THE USER-INDICATED BOUNDS. @THE ENTIRE FIELD IS PRINTED OUT AFTER EDIT. OPTIONAL COMMENTS

2
1132662
32662
32662
12662
32662
12662
11323621

2
1212362 7 73-80
03

10162 @PROGRAMMING@CONSIDERATIONS

01

- 13306621 1. ¢CUTE INPUT TEXT MAY BE MANUALLY OR TYPE-CODED ON FORM
2 ¢OCSPTC AVAILABLE THROUGH THE BOOKSTORE.
- 13306621 2. @TO MINIMIZE THE EDITING TIME SPENT ON INPUT TEXT PREPARATION IT IS RECOMMENDED THAT TEXTS BE PROCESSED IN PACKING MODE WHENEVER PRACTICABLE. @IN THIS MODE, ¢CUTE CAN PACK A PARAGRAPH USING INFORMATION PROVIDED ON THE FIRST AND LAST CARDS OF THE PARAGRAPH. @THE FIRST CARD PROVIDES THE LINE WIDTH AND SUBSEQUENTLY, LINES ARE PACKED, TRANSLATED AND RIGHT-JUSTIFIED UNTIL PACKING IS TURNED OFF BY THE LAST CARD OF THE PARAGRAPH. @IN PACKING MODE, PAGE-EJECT AND UNDERSCORE CARDS ARE RECOGNIZED AND PROPERLY BUFFERED. @CENTERING OR TRANSLATE-ONLY OPERATIONS ARE NOT RECOGNIZED, NOR DOES THE PROGRAM HEED ANY CHANGES IN@N1 OR@N2. @THE USER IS THEREFORE CAUTIONED TO BE SURE TO TURN OFF THE PACKING MODE AT THE END OF EACH PARAGRAPH.
- 2
13306621 3. @A CHARACTER IN COLUMN 10 OF INPUT CARD IS PRINTED IN PRINT POSITION 21 OF THE LINE PRINTER. @THIS INFORMATION WILL BE USEFUL IN ALIGNING THE EDGE OF SPECIAL OUTPUT FORMS.
- 2
13306621 4. @TEXTS FOR MULTIPLE REPRODUCTION SHOULD BE PRINTED ONTO A MAGNETIC TAPE WHICH CAN BE DUMPED REPEATEDLY BY A TAPE-TO-PRINT PROCESSOR.
- 2
13306621 5. @ERROR MESSAGES ARE RETURNED IN PLACE OF THE TEXT LINES THAT TRIGGER THEM. @THE MESSAGE IS PRECEDED AND FOLLOWED BY STARS.
- 2
1330662 6. @CORE REQUIREMENT FOR THE PROGRAM IS LESS THAN¢21K BYTES.
- 13306621 7. @THE FIRST PARAMETER CARD ALWAYS FORCES A PAGE-EJECT. @SUBSEQUENT PARAMETER CARDS DO NOT NECESSARILY FORCE A PAGE-EJECT.
- 2

13306621 8. @A LINE OF TEXT, AS IS PRINTED OUT, CONSISTS OF ANY
 2 CHARACTERS THAT ARE CONTAINED IN COLUMNS 9 THROUGH 72 OF
 13306621 9. @THE TRAILING BLANKS OF EACH LINE OF TEXT ARE NOT SIGNI-
 2 FICANT.
 13306621 10. @DURING THE RIGHT-MARGIN JUSTIFICATION OF A GROUP OF
 WORDS WITHIN A SPECIFIED FIELD, THE FIRST WORD WITHIN
 THE FIELD IS NOT DISPLACED UNLESS THE FIELD CONTAINS
 ONLY ONE WORD WHEREUPON THE WORD IS MOVED TO THE END OF
 THE FIELD. @THIS FACT, TOGETHER WITH THE MULTIPLE-EDIT
 FEATURE CAN BE USED TO MOVE A GROUP OF WORDS WITHIN A
 LINE OF TEXT. @THE USER SHOULD NOTE THAT THE ALPHABETIC-
 SHIFT-CONTROL CHARACTERS AND MUST BE ATTACHED TO THE
 01 WORDS
 01 THEY MODIFY; THE CONTROL CHARACTER MUST BE CONTIGUOUS
 2 WITH THE FIRST OF THE GROUP OF WORDS IT MODIFIES.
 13306621 11. @ONLY INTEGER NUMBERS MAY APPEAR IN COLUMNS 1 THROUGH 8
 OF ANY INPUT CARD. @FAILURE TO OBSERVE THIS RULE WILL
 TERMINATE THE JOB WITH THE SYSTEM@I/O ERROR MESSAGE@NO.
 @IHC215I. @A PARAMETER CARD, OTHER THAN THE
 2 VERY FIRST ONE, MUST BE IMMEDIATELY PRECEDED BY A CARD
 2 WHICH CONTAINS A THREE PUNCH IN COLUMN 8.
 2 \$ END OF FILE \$

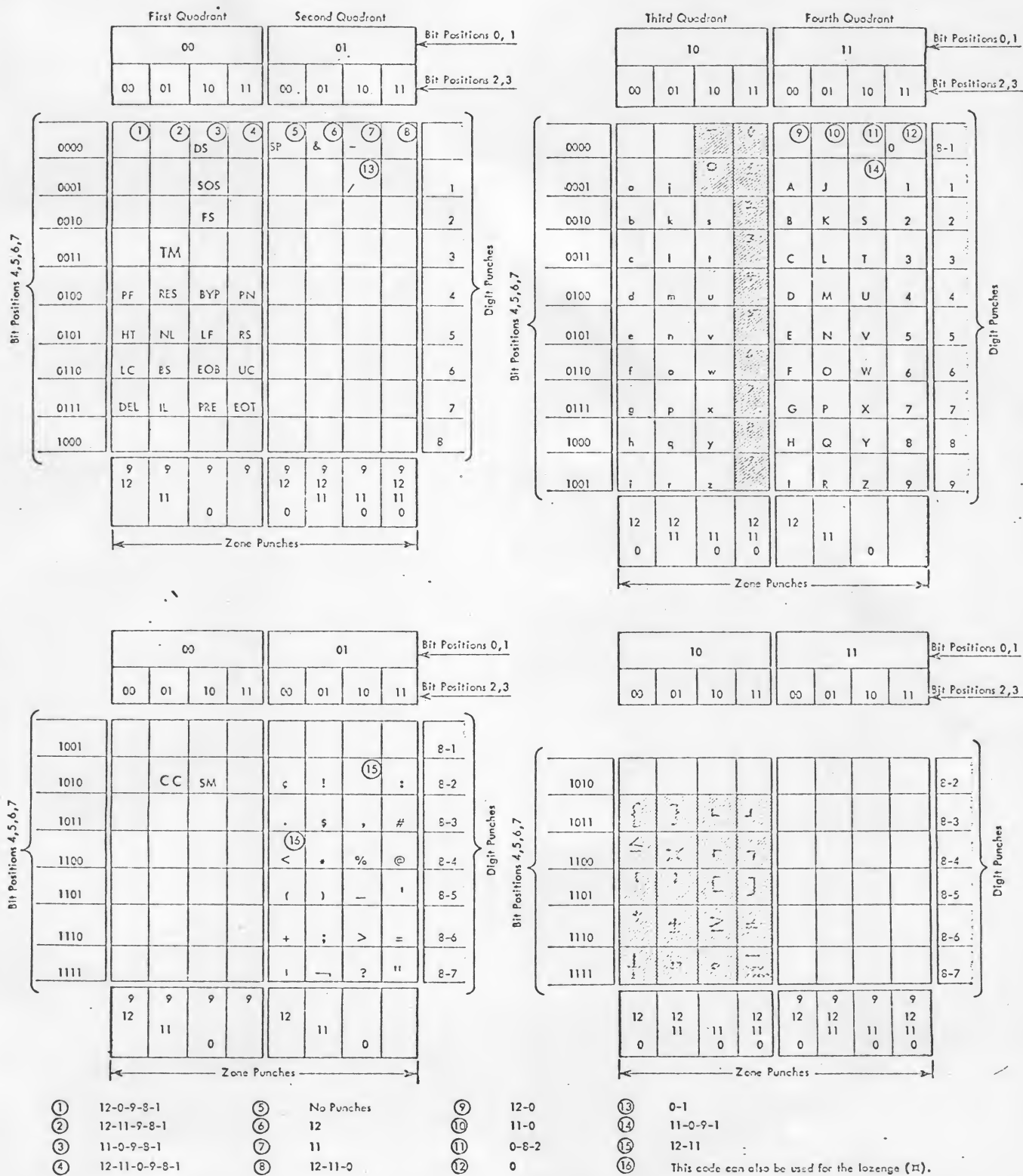


Figure 25. Extended Binary Coded Decimal Interchange Code

Copied From IBM Publication A24-3073-4

Markings on card edges (might be helpful)

Job No

AZHS61 - CUTE FORTRAN DRIVER
TEXEDT, TEXOUT

AZHS62 - CUTE SYSTEM:
CHARACTER MANIPULATION
THX
LINEDT - GH38
ASM. SOURCE 666 cards

AZHS63 - LIST CUTE MANUAL
SOURCE CARDS